Regular paper

Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms

Jaya Lakshmi Ravipudi, Mary Neebha

PII: S1434-8411(18)30170-5
DOI: https://doi.org/10.1016/j.aeue.2018.05.022
Reference: AEUE 52343

To appear in: *International Journal of Electronics and Communications*

Received Date: 21 January 2018
Accepted Date: 20 May 2018

Please cite this article as: J. Lakshmi Ravipudi, M. Neebha, Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms, *International Journal of Electronics and Communications* (2018), doi: https://doi.org/10.1016/j.aeue.2018.05.022

Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms

Jaya Lakshmi Ravipudi*and **Mary Neebha**
Department of Electronics and Communication Engineering
Karunya Institute of Technology and Sciences
Karunya Nagar, Coimbatore – 641114, Tamilnadu, India.
Phone: +919925207027
E-mail: ravipudijaya@gmail.com; maryneebha@karunya.edu
*Corresponding author
Jaya Lakshmi Ravipudi is a final year undergraduate engineering student.
Mary Neebha is holding a Masters degree in engineering and pursuing PhD program.

Abstract

This work aims to show the effectiveness of a recently proposed population-based optimization algorithm known as Jaya algorithm and its variants named as self-adaptive Jaya algorithm (SJaya) and Chaotic-Jaya (CJaya) algorithm to synthesize linear antenna arrays which are widely used in the communication systems. Three case studies of synthesis of linear antenna arrays are formulated by considering different topologies. In addition, two case studies of synthesis of dipole antenna arrays are formulated and all the case studies are solved using Jaya, SJaya and CJaya algorithms. The results of Jaya, SJaya and CJaya algorithms are compared with those of cat swarm optimization (CSO) algorithm, particle swarm optimization (PSO), Cauchy mutated cat swarm optimization (CMCSO) algorithm, harmony search based differential evolution algorithm (HSDEA), dynamic differential evolution algorithm (DDE), improved genetic algorithm (IGA), modified real genetic algorithm (MGA) and accelerated particle swarm optimization (APSO) algorithm. The Jaya, SJaya and CJaya algorithms achieved a better side lobe level suppression as compared to the other optimization algorithms while maintaining the vital antenna parameters within permissible limits.

Keywords: Linear antenna array synthesis; Optimization; Jaya algorithm; Self-adaptive Jaya; Chaotic-Jaya

1. Introduction

The wireless, satellite, mobile and radar communication systems use antenna arrays for better performance of the system. The use of antenna arrays help in enhancing directivity, improving signal quality, extending system coverage and increasing spectrum efficiency. The efficient design of the antenna arrays decides the performance of the communication system. The systems should have narrow first null beam width (FNBW) for obtaining high directivity and they need to maintain low sidelobe level (SLL) to avoid interference with other systems operating in the same frequency band. However, the arrays with lower side lobe levels may not produce narrow beam width and vice versa. Therefore, the SLL and FNB are mutually conflicting aspects of a linear antenna array as significant improvement in one aspect will degrade the other. It becomes necessary in many applications to sacrifice gain and beam width in order to achieve lower SLL. In addition, the placing of nulls in undesired directions is required to reduce the electromagnetic pollution. The structure of the array, distance between the elements and amplitude and phase excitation of individual elements affect the radiation pattern. A solution for suppressing SLLs and placing of nulls only in desired directions is to maintain optimal spacing between the element positions and maintaining uniform excitations. Another solution is to use non-uniform excitations of the elements with periodic placement of antenna elements.

The researchers have used a number of evolutionary algorithms for optimal design and synthesis of antenna arrays such as genetic algorithm (GA) [1-3], differential evolution (DE) [4,5], particle swarm optimization (PSO) [6,7], ant colony optimization (ACO) [8], invasive weed optimization (IWO) [9], bacteria foraging algorithm (BFA) [10], firefly algorithm (FA) [11], biogeography based optimization (BBO) [12], ant lion optimization (ALO) [13], cat swarm optimization (CSO) [14,15], etc.

Although, all the above mentioned evolutionary algorithms have shown the capability in electromagnetic optimization problems, the performance of these algorithms greatly depends upon their own algorithm-specific parameters. For example, GA requires tuning of selection operator, mutation probability, cross-over probability, etc.; PSO requires tuning of inertia weight and social and cognitive parameters, CSO requires tuning of mixing ratio, seeking memory pool, seeking range of dimension, inertia weight and acceleration coefficient. Similarly, other algorithms such as ACO, IWO, BFA, FA, BBO, ALO, etc. require tuning of their respective algorithm-specific parameters. If the control parameters of any algorithm are improperly tuned

the algorithm may converge at a local optimum or may slow down the algorithm's convergence rate. The tuning of algorithm-specific parameters is in addition to the common control parameters such as number of generations and population size. Therefore, before solving an optimization problem the user has to tune the algorithm-specific parameters of the algorithm along with the common control algorithms. For this purpose a number of trail runs with various combinations of algorithm-specific parameters are required to be conducted by the user. It is obvious that the user's effort increases with increase in the number of tunable parameters.

Rao [16] proposed Jaya algorithm which is simple in implementation and free from algorithm-specific parameters. The Jaya algorithm requires tuning of only the common control parameters such as number of generations or termination criteria and the population size. The Jaya algorithm has already proved its effectiveness in obtaining the global optimum for a number of constrained and unconstrained benchmark problems. Researchers have started applying the Jaya algorithm to solve a number of optimization problems [17-25].

Now in the present work the Jaya algorithm is applied, *for the first time,* in the field of electromagnetic engineering. In addition, an improved variant of the Jaya algorithm which based on the Chaos theory is proposed in this paper and it is named as the Chaotic-Jaya (CJaya) algorithm. Another improved version of Jaya algorithm named as, self-adaptive Jaya algorithm [22,23], is also considered for the purpose of comparison of results. In this paper the Jaya, self-adaptive Jaya and Chaotic-Jaya algorithms are used to obtain the optimum spacing between the elements of the linear antenna array and dipole antenna array to produce a radiation pattern with minimum SLLs. The codes for Jaya, self-adaptive Jaya and Chaotic-Jaya algorithms are developed using MATLAB R2014R and a computing platform with Intel Core i5 CPU 2.30 GHz processor with 4 GB RAM is used for execution of the codes.

This paper is prepared as follows. Section 2 presents brief description of Jaya, self-adaptive Jaya and Chaotic-Jaya algorithms. The linear array configuration is discussed in Section 3.1. Three case studies on synthesis of linear antenna arrays with different topologies are formulated in section 3.1 and the results are discussed.The configuration of the dipole antenna is discussed in Section 3.2 and two case studies are formulated and the results are discussed. Section 4 presents the conclusions.

2. Jaya, self-adaptive Jaya and chaotic-Jaya algorithms

2.1 Jaya algorithm

In Jaya algorithm P initial solutions are randomly generated keeping in view of the upper and lower range values of the design variables. Subsequently, the values of variables of every solution are updated using Eq. (1). Let f is the objective function to be minimized (or maximized). Let there be 'd' number of design variables. Let the best solution be *f_best* and the worst solution be *f_worst* [16].

$$A(i+1,j,k)=A(i,j,k)+r(i,j,1)(A(i,j,b)-|A(i,j,k)|)-r(i,j,2)(A(i,j,w)-|A(i,j,k)|) \quad (1)$$

Here b and w represent the indices of the best and worst solutions among the population. i, j, k are the indices of iteration, variable, and the candidate solution respectively. $A(i, j, k)$ means the j^{th} variable of k^{th} candidate solution in i^{th} iteration. $r(i, j, 1)$ and $r(i, j, 2)$ are the randomly generated numbers in the range of [0, 1]. The random numbers $r(i, j, 1)$ and $r(i, j, 2)$ act as scaling factors and ensure good diversification. The flowchart of Jaya algorithm is shown in Fig. 1.

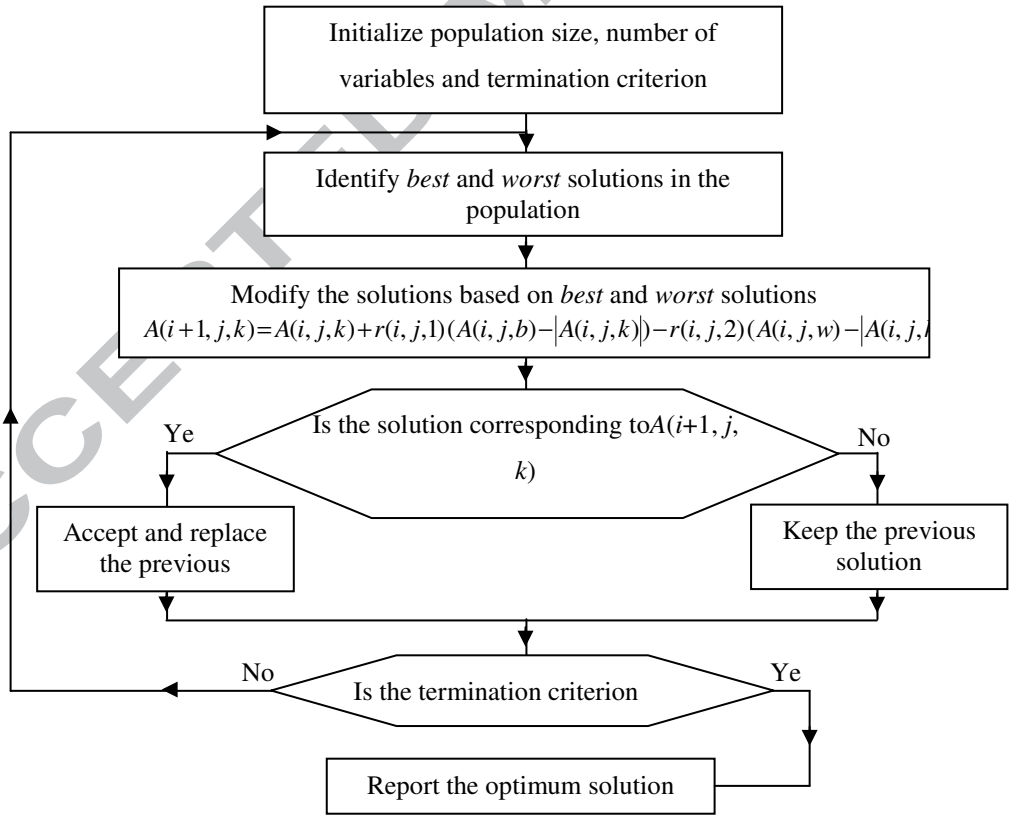

Fig. 1.

The Jaya algorithm improves the objective function value of each candidate solution in the population. The algorithm tries to move the objective function value of each solution towards the best solution by updating the values of the variables. Once the values of the variables are updated, the same are compared with the corresponding old solutions and only the solutions having better objective function value are considered for the next generation [16].

2.2 Self-adaptive Jaya algorithm

Rao and More [22,23] proposed a self-adaptive Jaya (SJaya) algorithm. The key feature is that the self-adaptive Jaya algorithm determines the population size automatically. Hence, the user need not concentrate on choosing the population size. Let the random initial population is (10**l*), where *l* is the number of design variables then the new population is formulated as,

$$n_{new} = \text{round}\ (n_{old} + r^{*} n_{old}) \quad (3)$$

where r is a random value between [0.5, 0.5]; it acts as a comparative population development rate. The population size may decrease or increase due to negative or positive random value of r. Elitism is implemented when the population size of the next iterations is larger than the population size of the present generation ($n_{new} > n_{old}$). Then all of the existing population will go into the next generation and the best optimal solutions in the current population are assigned to the remaining nnewnold solutions. When the population size of the next generation is less than the population size of the current generation ($n_{new} < n_{old}$), then, only the best population of the current iteration will go to the next generation. No change takes place if the population sizes of the next and current generations are equal ($n_{new} = n_{old}$). If the population size decreases and becomes less than the number of design variables (l), then the population size is considered equal to the number of design variables (if $n_{new} < l$, then $n_{new} = l$).

2.3 Chaotic Jaya algorithm

A variant of Jaya algorithm is proposed in this work which is based on the chaos theory and is named as the Chaotic-Jaya (CJaya) algorithm. This is to improve the convergence speed and to better explore the search space without entrapment into local optima. Mathematically, chaos is randomness of a simple deterministic dynamical system and chaotic system may be considered as source of randomness. In order to introduce chaos in optimization algorithms, different chaotic maps having different mathematical equations are used. Since last decade,

chaotic maps have been widely appreciated in the field of optimization due to their dynamic behavior which help optimization algorithms in exploring the search space more dynamically and globally. A wide variety of chaotic maps designed by physicians, researchers and mathematicians are available in the optimization field [26]. In these chaotic maps, any number in the range [0,1] (or according to the range of chaotic map) may be chosen as the initial value. However, the initial value may have significant impacts on the fluctuation pattern of some of the chaotic maps. This set of chaotic maps has been chosen with different behaviors, while the initial value is 0.7 for all [27]. Chaotic maps are believed to affect the convergence rate of Jaya algorithm positively as these maps induce chaos in the feasible region which is predictable only for very short initial time and is stochastic for longer period of time.

The working of the Chaotic-Jaya (CJaya) algorithm is exactly similar to the Jaya algorithm, the only difference is that in CJaya algorithm, the random numbers are produced using a chaotic random number generator. In this work the tent map chaotic function is used as the chaotic random number generator because of its simplicity and it is expressed by Eq. (2).

$$x_{k+1} = \begin{Bmatrix} \dfrac{x_k}{0.7} & x_k < 0.7 \\ \dfrac{10}{3}(1 - x_k) & x_k \geq 0.7 \end{Bmatrix} \tag{2}$$

Where, x_{k+1} is the newly generated chaotic random number and x_k is the previously generated chaotic random number.

3 Case studies

3.1 Synthesis of linear antenna array

In this paper, three optimization case studies have been formulated for synthesis of 28-elements and 37-elements linear antenna arrays. The Jaya algorithm is applied to each of the case studies to achieve lowest peak SLL. The optimization case studies are divided into two categories. In the first category, minimization of peak SLL is considered without the constraint of first null beam width (FNBW). In the second category, minimization of peak SLL is considered while maintaining the FNBW as constant.

Fig. 2 shows the geometry of a uniform linear antenna array with 2N elements placed symmetrically along *x*-axis. The array factor (AF) in the azimuth plane is calculated as follows.

$$AF(\theta) = 2\sum_{n=1}^{N} I_n \cos\left[kx_n \cos(\theta) + \varphi_n\right] \quad (4)$$

Where $k = 2\pi/\lambda$ is the wave number, θ is the azimuth angle, φ_n, x_n and I_n are the phase, position of element n, and excitation amplitude, respectively. Uniform amplitude and phase excitation are assumed for all the elements i.e. $I_n = 1$ and $\varphi_n = 1$ [14].

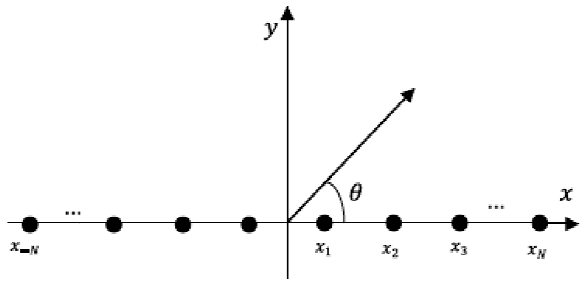

Fig. 2.

Fig. 3 shows the geometry of uniform linear antenna array with odd number of elements 2N+1 (one element is placed at the centre) symmetrically placed along x-axis. The e array factor (AF) in the azimuth plane is expressed by Eq. (5).

$$AF(\theta) = 1 + 2\sum_{n=1}^{N} I_n \cos\left[kx_n \cos(\theta) + \varphi_n\right] \quad (5)$$

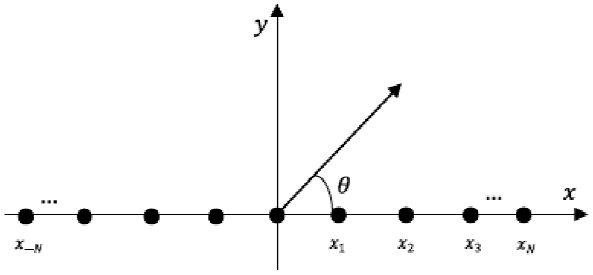

Fig. 3.

In the case of linear antenna array the aim is to suppress the peak SLL in a specific region. For this purpose an objective function is expressed by Eq. (6) [14].

$$Fitness = min\left[max\left(20log\frac{|AF(\theta)|}{max|AF(\theta)|}\right)\right] \quad (6)$$

In antenna arrays, placement of the adjacent elements very close leads to mutual coupling effects and placement too far leads to grating lobes. The following conditions are imposed on placement of elements in linear array antenna [14].

$$\left|x_i - x_j\right| > 0.25\lambda \tag{7}$$

$$\min\{x_i\} > 0.125\lambda \quad i = 1,2,.....,N. \quad i \neq j \tag{8}$$

The above two constraints which are imposed to prevent mutual coupling effects and grating of lobes are considered in all seven case studies.

Case study 1: This case study presents the synthesis of a 28-element linear antenna array for minimization of peak side lobe level in the region of θ = [0°,74°] and θ = [106°,180°] with varying FNBW. This problem was solved by Pappula and Ghosh [14] using CSO algorithm with maximum number of function evaluations of 50000 to obtain the best result. Now, the same problem is solved using Jaya and CJaya algorithms with the same maximum number of function evaluations for fair comparison of results. The optimized array patterns, SLL and FNBW obtained by Jaya, CJaya, SJaya, CSO and PSO algorithms for minimizing SLL for 28-element array with varying FNBW is shown in Table 1.

It is observed that the minimum SLL obtained by the optimized array pattern provided by CJaya algorithm is -32.459 dB which is better than the minimum SLL obtained by conventional array pattern, PSO optimized array pattern and CSO optimized array pattern. The CJaya algorithm has minimized the SLL from -13.27 dB to -32.459 dB (reduced by 19.189 dB) compared to the conventional array pattern and from -21.89 dB to -32.459 dB (reduced by 10.569 dB) as compared to the array pattern provided by PSO algorithm. The CJaya algorithm has minimized the SLL from -24.53 dB to -32.459 dB (reduced by 7.929 dB) as compared to the SLL obtained by the CSO algorithm [14]. The CJaya algorithm has minimized the SLL from -30.129 dB to -32.459 dB (reduced by 2.33 dB) as compared to the SLL obtained by the Jaya algorithm. The CJaya algorithm has minimized the SLL from -30.205 dB to -32.459 dB (reduced by 2.254 dB) as compared to the SLL obtained by the SJaya algorithm.

Table 1

Optimized positions, SLL and FNBW obtained by Jaya, CJaya,, SJaya, CSO and PSO algorithms for minimizing SLL for 28-element array with varying FNBW (case study 1).

Algorithm	Optimized element positions (x/λ)							Optimized	
								SLL	FNBW
PSO	0.1703	0.6430	0.9509	1.4245	1.7849	2.0397	2.4511	-21.89	10^o
	3.0522	3.6249	4.0476	4.6302	5.2984	5.9582	6.7118		
CSO	0.2344	0.5280	0.9224	1.2965	1.6549	2.1427	2.4387	-24.53	10.5^o
	2.9369	3.3753	3.9280	4.4091	5.1167	5.9188	6.7422		
Jaya	5.9860	5.3424	4.9259	4.4249	4.1749	3.6524	3.3832	-30.129	9.4^o
	2.8526	2.4617	1.9100	1.4956	0.9647	0.6504	0.1250		
CJaya	0.2162	0.5855	1.0706	1.636	2.3196	2.8794	3.3600	**-32.459**	8.8^o
	3.6528	4.0495	4.3479	4.5980	5.0474	5.3372	5.9687		
SJaya	0.4152	1.1446	1.6626	2.1387	2.4671	2.8324	3.1709	-30.205	8.8^o
	3.4211	3.7617	4.1006	4.3519	4.8007	5.1867	5.8220		

The results of PSO and CSO algorithms are reproduced from Pappula and Ghosh [14]. Values in bold indicate better performance of the algorithm.

Furthermore, Table 1 shows that the CJaya algorithm has provided the lowest value of FNBW as compared PSO, CSO and Jaya algorithms. The CJaya algorithm has reduced the FNBW from 10^o to 8.8^o (reduced by 1.2^o) as compared to the FNBW provided by PSO algorithm and from 10.5^o to 8.8^o (reduced by 1.7^o) as compared to the FNBW provided by CSO algorithm. The CJaya algorithm has reduced the FNBW from 9.4^o to 8.8^o (reduced by 1.2^o) as compared to the FNBW provided by Jaya algorithm. The CJaya and SJaya algorithms have obtained the same FNBW value of 8.8^o.

The comparison of radiation patterns obtained by PSO, CSO, Jaya, SJaya and CJaya algorithms of 28-element linear array synthesis with varying FNBW is shown in Fig. 4. The convergence of Jaya, SJaya and CJaya algorithms for 28-element linear array with varying FNBW is shown in Fig. 5. It is observed from Fig. 5 that the Jaya algorithm required about 1550 generations for convergence. On the other hand, SJaya required about 550 generations and CJaya required about 900 generations for convergence. Even though CJaya algorithm has converged at

about 900 generations, the minimum value of SLL obtained by CJaya algorithm is better than that obtained by SJaya algorithm. The actual number of generations for convergence and the time required by CSO and PSO algorithms is not reported by Pappula and Ghosh [14]. The time taken by Jaya, CJaya and SJaya algorithms is 48.966447s, 47.391925s, and 55.117286s respectively.

It may be noted that the computational time that we have considered is not the time required by the algorithms to converge to its best solution, rather, it is the time required by the Jaya, CJaya and SJaya algorithms to complete the maximum number of function evaluations set by the user (i.e. 50000 in this case). Therefore, even though an algorithm may have converged, the algorithm will continue to execute until the maximum number of function evaluations are reached. Even after convergence, the algorithm has to execute steps such as generating new solutions (i.e. updation), comparing the new solutions with the previous solutions (i.e. selection), etc, until the termination criterion is satisfied (i.e. maximum number of function evaluations of 50000 in this case).

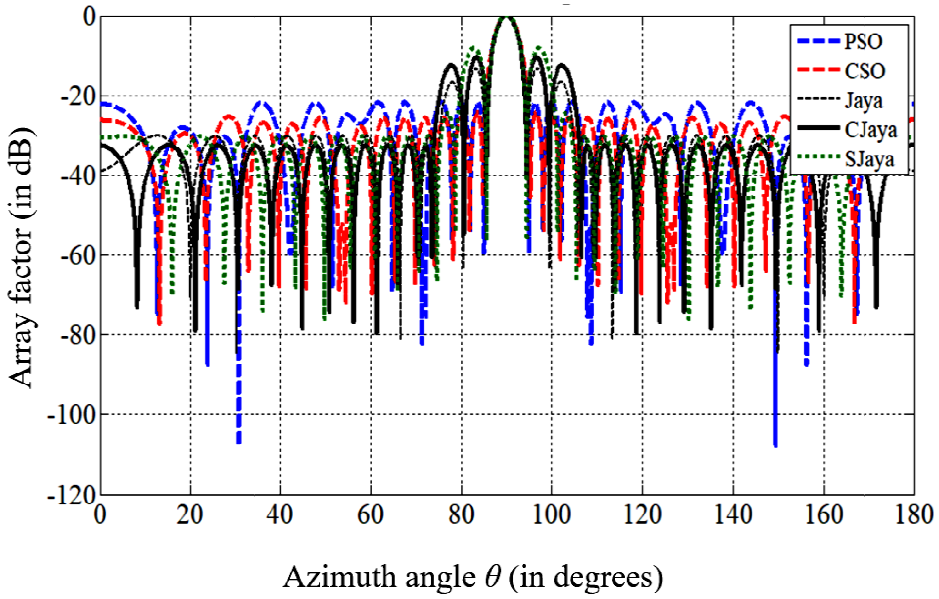

Fig. 4.

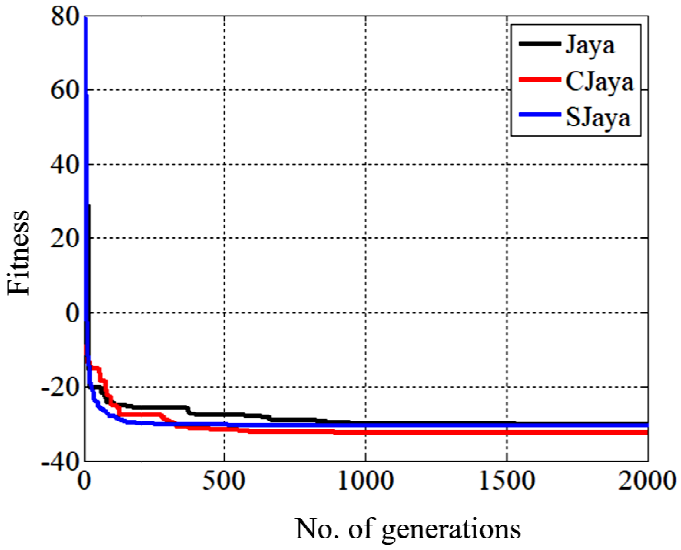

Fig. 5.

Case study 2: In this example, a 28-element linear antenna array is synthesized for obtaining minimum value of SLL. In addition to the constraints expressed by Eq. (7) and Eq. (8) it is desired to maintain the FNBW as constant at 8° with a beam width tolerance set at ±5%. This problem was solved by Pappula and Ghosh [14] using CSO algorithm with maximum number of function evaluations of 50000 to obtain the best result. Now, the same problem is solved using Jaya, SJaya, and CJaya algorithms with the same maximum number of function evaluations for fair comparison of results.

The optimized array patterns, SLL and FNBW obtained by Jaya, SJaya, CJaya, CSO and PSO algorithms for minimizing SLL for 28-element array with constant FNBW is shown in Table 2. The SJaya algorithm has suppressed the SLL from -13.27 dB to -31.954 dB (reduced by 18.684 dB) as compared to the SLL obtained by the conventional array pattern; and from -17.22 dB to -31.954 dB (reduced by 14.734 dB) as compared to the SLL obtained by the PSO optimized array pattern. The SJaya algorithm has suppressed the SLL from -20.32 dB to -31.954 dB (reduced by 11.634 dB) as compared to the SLL obtained by CSO optimized array pattern. The performance of SJaya algorithm is found better than the Jaya and CJaya algorithms in this case study.

Table 2

Optimized positions, SLL and FNBW obtained by Jaya, CJaya, SJaya, CSO and PSO algorithms for minimizing SLL for 28-element array with constant FNBW (case study 2).

Algorithm	Optimized element positions (x/λ)							Optimized	
								SLL	FNBW
PSO	0.3270	0.5771	0.2089	1.5145	2.1417	2.3939	2.8792	-17.22	8.1^o
	3.4450	4.3046	4.8928	5.1472	5.9070	6.4275	6.9999		
CSO	0.2437	0.6445	1.0230	1.5095	1.8444	2.3974	2.8835	-20.32	8.1^o
	3.2657	3.8500	4.4726	5.1068	5.8367	6.5065	6.9999		
Jaya	3.3257	0.4198	4.2421	3.1259	4.9242	2.2649	5.3105	-22.886	**8^o**
	1.6476	5.9642	4.4748	1.0641	3.6378	2.5849	3.8806		
CJaya	2.2229	0.4656	5.9410	4.7940	3.9154	5.2783	1.5966	-27.73	**8^o**
	2.2601	3.5381	6.7147	1.2198	3.0694	4.3103	2.8957		
SJaya	0.1907	0.6148	1.0839	1.7375	2.4846	3.0434	3.5114	**-31.954**	7.9^o
	3.7359	4.2551	4.3160	4.7288	5.0888	5.4047	6		

The results of PSO and CSO algorithms are reproduced from Pappula and Ghosh [14]. Values in bold indicate better performance of the algorithm

Fig. 6 shows the radiation patterns obtained by PSO, CSO, Jaya, SJaya and CJaya algorithms for 28-element linear array synthesis with constant FNBW. The convergence of Jaya SJaya and CJaya algorithms for 28-element linear array with constant FNBW is shown in Fig. 7. It is observed from Fig. 7 that the Jaya algorithm required about 1400 generations, CJaya algorithm required about 450 generations and SJaya algorithm required about 1900 generations to converge. The time taken by Jaya, CJaya and SJaya algorithms is 50.62672s, 49.836186s and 53.343088s respectively. The actual number of generations for convergence and the time required by CSO and PSO algorithms is not reported by Pappula and Ghosh [14].

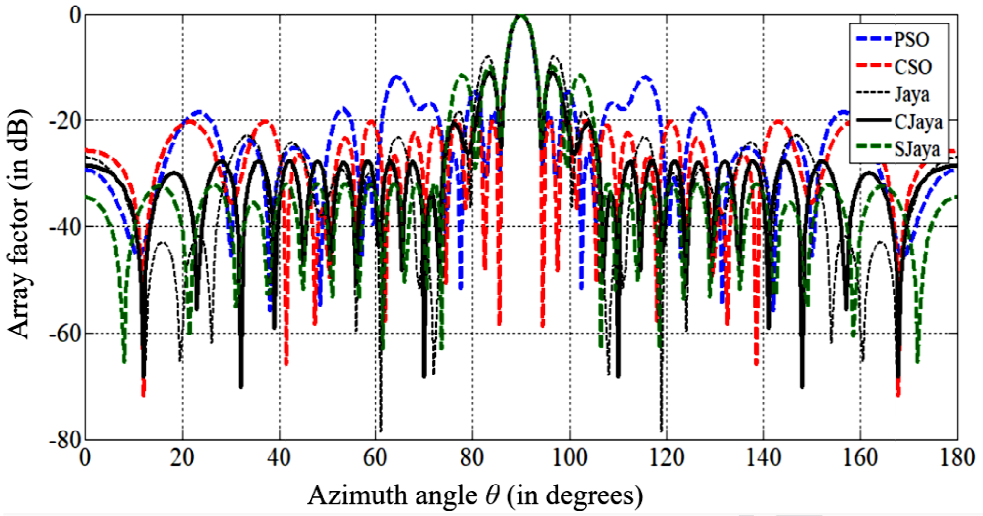

Fig. 6.

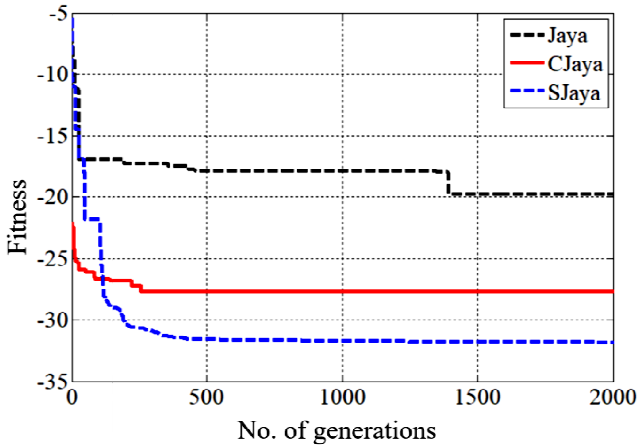

Fig. 7.

Case study 3: This case study presents the synthesis of a 37-element array for minimization of SLL with varying FNBW. The number of elements in the array are odd and hence the array factor is evaluated according to Eq. (5). The objective function to be minimized is expressed by Eq. (6) and the constraints given by Eqs. (7) and (8) are considered. The same problem was solved by Pappula and Ghosh [15] using Cauchy mutated cat swarm optimization (CMCSO) algorithm. Maximum number of generations of 2000 and population size of 50 (i.e. maximum number of function evaluations of 2000 x 50 = 100000) were considered for CMCSO algorithm. Now, the Jaya, SJaya and CJaya algorithms are applied to solve the same problem maintaining the same conditions of function evaluations of 100000 for fair comparison using maximum number of generations of 5000 and population size of 20.

The optimized array pattern obtained by Jaya, SJaya and CJaya algorithms is shown in Table 3. The Jaya, SJaya and CJaya algorithms are executed 20 times independently and comparison of the results is made with the results obtained by CMCSO algorithm, differential evolution algorithm based on harmony search (HSDEA), dynamic differential evolution (DDE) algorithm, improved genetic algorithm (IGA) and modified real genetic algorithm (MGA) on the basis of best SLL, average SLL and average number of function evaluations required by each algorithm to achieve convergence over 20 independent runs.

Table 3

Optimized array patterns obtained by Jaya, CJaya and SJaya algorithms for 37-element linear aperiodic array synthesis (case study 3).

Element (n)	Position (x_n/λ)			Element (n)	Position (x_n/λ)		
	Jaya	CJaya	SJaya		Jaya	CJaya	SJaya
1	0.778080	0.7543	0.7462	10	5.40480	5.7885	5.5039
2	1.286757	1.5221	1.4998	11	6.12576	6.1481	5.7418
3	1.769949	2.3056	2.3205	12	6.35244	6.448	6.2209
4	2.026993	3.0904	3.0850	13	6.88425	6.6164	6.5818
5	2.535492	3.8107	3.6662	14	7.10913	7.1381	7.2939
6	2.718854	4.3678	4.1293	15	7.46707	7.1429	8.2037
7	3.227185	4.8433	4.5074	16	7.86860	7.6889	9.1259
8	3.713001	5.1694	4.8102	17	8.17316	8.0310	9.9784
9	4.521139	5.5879	5.1319	18	8.79261	8.6614	10.7960

Table 4 presents the results of Jaya, SJaya and CJaya algorithms along with the results of other algorithms like CMCSO [15], HSDEA [3], DDE [4,5], IGA [2] and MGA [3]. The Jaya algorithm has suppressed the SLL from -20.6 dB to -30.399 dB (reduced by 9.799 dB) and from -22.4 dB to -30.399 dB (reduced by 7.999 dB) as compared to the array pattern obtained by MGA and IGA algorithms, respectively. The Jaya algorithm has suppressed the SLL from -22.62 dB to -30.399 dB (reduced by 7.779 dB) and from -22.59 dB to -30.399 dB (reduced by 7.809 dB) and compared to SLL suppression obtained by DDE and HSDEA algorithms, respectively.

The Jaya algorithm has suppressed the SLL from -22.95 dB to -30.399 dB (reduced by 7.449 dB) as compared to the SLL suppression obtained by CMCSO algorithm.

Table 4

Comparison of SLL obtained by Jaya, CJaya and SJaya algorithms for 37-element linear aperiodic array synthesis with the SLL obtained by other algorithms (case study 3).

SLL (dB)	SJaya	CJaya	Jaya	CMCSO	HSDEA	DDE	IGA	MGA
Best	**-**29.1177	**-32.9246**	-30.399	-22.95	-22.59	-22.62	-22.4	-20.6
Average	-24.4568	**-29.866**	-27.854	-22.64	NA	-22.59	-22.10	-20.3
Average FEs	31018	32053	52469	48450	NA	99607	10186	60000

NA: not available in the literature. Values in bold indicate better performance of the algorithm as compared to the other algorithms. The results of CMCSO, HSDEA, DDE, IGA and MGA algorithms are reproduced from Pappula and Ghosh [15].

It is observed that, the CJaya algorithm has suppressed the SLL from -20.6 dB to -32.9246 dB (reduced by 12.3246 dB) as compared to the array pattern obtained by MGA and from -22.4 dB to -32.9246 dB (reduced by 10.5246 dB) as compared to the array pattern obtained by IGA algorithms. The CJaya algorithm has suppressed the SLL from -22.62 dB to -32.9246 dB (reduced by 10.3046 dB) and from -22.59 dB to -32.9246 dB (reduced by 10.334 dB) and compared to SLL suppression obtained by DDE and HSDEA algorithms, respectively. The CJaya algorithm has suppressed the SLL from -22.95 dB to -32.9246 dB (reduced by 9.9746 dB) as compared to the SLL suppression obtained by CMCSO algorithm.

The CJaya algorithm has suppressed the SLL from -30.399 dB to -32.9246 dB (reduced by 2.5256 dB) as compared to the SLL suppression obtained by Jaya algorithm. The CJaya algorithm has suppressed the SLL from -29.1177 dB to -32.9246 dB (reduced by 3.8069 dB) as compared to the SLL suppression obtained by SJaya algorithm. The Jaya algorithm required an average number of function evaluations (FEs) of 52469 for convergences over 20 independent runs. On the other hand, the CJaya algorithm required 32053 FEs and SJaya algorithm required 31018 FEs. The comparison of radiation patterns obtained by Jaya, SJaya, CJaya and CMCSO algorithms of 37-element linear aperiodic array synthesis is shown in Fig. 8. The convergence of Jaya, SJaya and CJaya algorithms for 37 element linear array with varying FNBW is not shown here for space reasons. The time taken by Jaya, CJaya and SJaya algorithms is 113.077425 s.,

103.937359 s.and 109.290842 s respectively. The actual number of generations for convergence and the time required by CMCSO, HSDEA, DDE, IGA and MGA algorithms is not reported by Pappula and Ghosh [15].

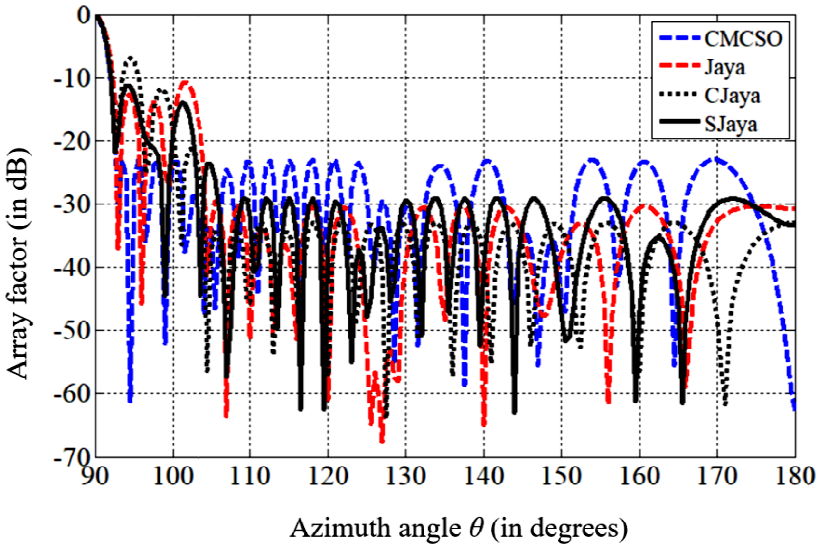

Fig. 8.

3.2 Synthesis of dipole array antenna

In this illustration, an array of 2N number of identical half-wavelength dipoles spaced at a distance along the z-axis is considered. The geometry and excitation are assumed symmetric with respect to the centre of the array. The far field $AF(\theta)$ pattern in the horizontal x-y plane, F(θ) is expressed as follows:

$$F(\theta)=\frac{\cos(0.5\pi\cos(\theta))}{\sin(\theta)} \tag{9}$$

$$AF(\theta)=\sum_{n=1}^{N}2I_n\cos(kd_n\cos(\theta))\times F(\theta) \tag{10}$$

The element pattern is calculated using Eq. (11). The normalized array pattern in dB is given as:

$$AF_n(\theta)=20\log_{10}\left[\frac{|AF(\theta)|}{|AF(\theta)|_{\max}}\right] \tag{11}$$

Here, n is the element number, k =2π/λ is the wave number, λ is the wavelength, d_n is the distance of centre of the n^{th} element from origin, N is the total number of elements, θ is the polar angle of the far field measured from broad side.

The conditions of low SLL and optimum dynamic range ratio (DRR) are to be satisfied by the radiation pattern produced by the array. DRR is the ratio of maximum excitation amplitude (I_{max}) to the minimum excitation amplitude (I_{min}).It is expected to obtain an optimal placement of array elements and excitation amplitudes such that the SLL is minimized at the same time the obtained DRR value must be closest to the desired DRR (DRR_{des} = 7.0). Therefore, an objective function is formulated, accordingly and is expressed by Eq. (12).

$$Minmize\ fitness = a \times \{AF_n(\theta)\} + b \times (DRR - DRR_{des})^2 \tag{12}$$

Where, a and b are the weights. $AF_n(\theta)$ is the normalised array factor.

Case study 4: This case study illustrates the synthesis of a 12-element dipole antenna for minimization of SLL with varying FNBW with an additional objective to minimize the difference between the achieved DRR and DRR_{des} value. The objective function expressed by Eq. (12) is considered with n = 12. Florence and Raju [28] minimized the objective function using accelerated particle swarm optimization (APSO) algorithm with population size of 20 and maximum number of iterations of 100 (i.e. maximum number of function evaluations of 2000). Now the same problem is solved using Jaya, SJaya and CJaya algorithms using the same number of function evaluations to check for any further improvement. Table 5 shows the positions of array elements (x) and the excitement amplitude (I) obtained by Jaya, SJaya and CJaya algorithms for a dipole antenna with 12 array elements.

Table 6 shows the values of SLL, FNBW, DRR and the value of objective function corresponding to the optimized array positions and amplitude obtained by the APSO, Jaya, SJaya and CJaya algorithms for a dipole antenna with 12 array elements.

The results show that the APSO algorithm has obtained the SLL equal to -37.41 dB. The Jaya algorithm minimized the SLL by 2.12 %, from -37.41 dB to -37.81 dB as compared to the SLL obtained by APSO algorithm. The DRR value obtained by the Jaya algorithm is closer to the DRR_{des} as compared to the DRR value obtained by APSO algorithm. The fitness function value obtained by Jaya algorithm is -18.9026 and is lower than the fitness value of -18.51 given by APSO algorithm.

Table 5

Excitation amplitude and spacing obtained by Jaya, SJaya and CJaya algorithms for 12 element antenna array (case study 4).

Element number	Spacing				Position (x/λ)				Amplitude (I)		
	Jaya	CJaya	SJaya		Jaya	CJaya	SJaya		Jaya	CJaya	SJaya
1	0.200	0.2000	0.2000		0.2	0.2	0.2		0.9481	0.4120	0.8249
2	0.5616	0.4048	0.5896		0.7616	0.6048	0.7896		0.9703	0.6037	1.000
3	0.4042	0.6744	0.4458		1.1658	1.2792	1.2354		0.5957	0.6109	0.3091
4	0.5697	0.7233	0.4000		1.7355	2.0025	1.6354		0.6963	0.3288	0.6680
5	0.5048	0.5039	0.7103		2.2403	2.5064	2.3457		0.1400	0.1056	0.4109
6	0.4003	0.5059	0.7650		2.6406	3.0123	3.1107		0.2479	0.0867	0.1447

Table 6

The values of SLL, FNBW, DRR and fitness obtained by APSO, Jaya, CJaya and SJaya algorithms for 12 element dipole antenna (case study 4).

N	Algorithm	SLL	FNBW	DRR	*fitness*
12	APSO	-37.41	29.96	6.37	-18.5100
	Jaya	-37.81	33.599	6.9307	-18.9026
	CJaya	-43.2286	33.20	**7.05**	-21.6131
	SJaya	**-45.8964**	**33**	6.909	**-22.94413**

The results show that, as compared to the APSO algorithm, the CJaya algorithm reduced the SLL by 15.55%, from -37.41 dB to -43.2286dB.The DRR value obtained by the CJaya algorithm is 7.05 which is closer to the DRR_{des} as compared to the DRR value obtained by APSO algorithm. The FNBW value reported by CJaya algorithm is better compared APSO algorithm. The *fitness* value obtained by Jaya algorithm is 16.76 % better than that obtained by APSO.

The CJaya algorithm showed better performance compared to Jaya algorithm in terms of SLL, DRR and FNBW. The CJaya algorithm minimized the SLL by 14.33% as compared to the SLL given by Jaya algorithm. The CJaya algorithm obtained better fitness value compared to

Jaya algorithm and the DRR value obtained is more precise. The SJaya algorithm showed better performance compared to CJaya algorithm in terms of SLL, FNBW and the fitness.

The radiation patterns obtained by APSO, Jaya, SJaya and CJaya algorithms is shown in Fig. 9. The convergence of Jaya, SJaya and CJaya algorithms is not shown here for space reasons. Jaya and CJaya algorithms have converged at about 140 generations and SJaya at about 160 generations for this case study. The actual number of generations for convergence and the time required by APSO was not reported by Florence and Raju [28]. The time taken by Jaya, CJaya and SJaya algorithms is 57.488s, 53.980004s and 55.98426s respectively.

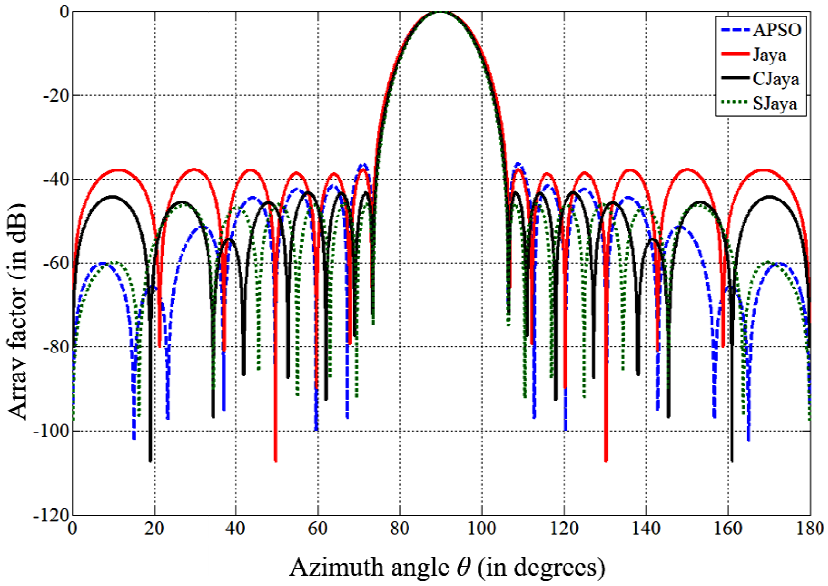

Fig. 9.

Case study 5: This case study illustrates the synthesis of a 20-element dipole antenna for minimization of SLL with varying FNBW with a minimum difference between the achieved DRR and DRR_{des} values. The objective function expressed by Eq. (12) is considered with $n = 20$. Florence and Raju [28] solved this problem using APSO algorithm considering maximum number of function evaluations of 2000. The same problem is now solved using Jaya, SJaya and CJaya algorithms with the same number of function evaluations. Table 7 shows the optimized positions of array elements (x) and the excitement amplitude (I) obtained by Jaya, SJaya and CJaya algorithms for a dipole antenna with 20 array elements. Table 8 shows the values of SLL, FNBW, DRR and the value of fitness function corresponding to the optimized array positions and amplitude obtained by the APSO, Jaya, SJaya and CJaya algorithms for a dipole antenna with 20 elements.

Table 7

Excitation amplitude and spacing obtained by Jaya, CJaya, SJaya algorithms for 20 element dipole antenna (case study 5).

Element number	Spacing				Position				Amplitude		
	Jaya	CJaya	SJaya		Jaya	CJaya	SJaya		Jaya	CJaya	SJaya
1	0.2000	0.2000	0.2000		0.2	0.2	0.2		0.2481	0.9992	0.4360
2	0.4003	0.4051	0.4252		0.6003	0.6051	0.6252		0.4461	1.000	0.7584
3	0.6218	0.4151	0.7365		1.2221	1.0202	1.3617		0.6292	1.000	0.9894
4	0.5979	0.4007	0.8000		1.82	1.4209	2.1617		0.7715	0.9210	0.9390
5	0.6577	0.4296	0.4016		2.4777	1.8505	2.5633		0.9933	0.9968	0.1797
6	0.7222	0.4781	0.4908		3.1999	2.3286	3.0541		0.9732	0.9861	0.9213
7	0.6883	0.5767	0.7985		3.8882	2.9053	3.8526		0.6108	1.000	0.82802
8	0.4093	0.6888	0.7998		4.2975	3.5941	4.6524		0.2951	0.7546	0.60307
9	0.5686	0.7446	0.7981		4.8661	4.3387	5.4505		0.3539	0.4018	0.3405
10	0.7722	0.7885	0.7924		5.6383	5.1272	6.2429		0.1386	0.1380	0.1319

Table 8

The values of SLL, FNBW, DRR and fitness obtained by APSO, Jaya, CJaya and SJaya algorithms for 20 element dipole antenna (case study 5).

N	Algorithm	SLL	FNBW	DRR	*fitness*
20	APSO	-38.89	18.39	6.84	-19.4390
	Jaya	-49.7907	14.19	**7.16**	-24.881
	CJaya	**-51.532**	17.199	7.24	**-25.7364**
	SJaya	-48.0882	11.600	7.4992	-23.9195

The results show that the APSO algorithm has obtained SLL equal to -38.89 dB. However, the Jaya algorithm reduced the SLL by 15.35 %, from -38.89 dB to -44.861 dB.The CJaya algorithm reduced the SLL by 32.50 %, from -38.89 dB (which was obtained by APSO algorithm) to -51.532 dB. The CJaya algorithm reduced the SLL by 7.16 %, from -48.0882 dB (which was obtained by SJaya algorithm) to -51.532 dB. The DRR value obtained by the Jaya

algorithm is 7.16 which is much closer to the DRR_{des}. The CJaya and Jaya algorithms obtained a better value of FNBW as compared APSO and SJaya algorithms. The *fitness* values obtained by Jaya and CJaya algorithms are better than those given by APSO and SJaya algorithms for this case study. CJaya algorithm has obtained better fitness value compared to all other algorithms. Fig. 10 shows the radiation patterns obtained by Jaya, SJaya, CJaya and APSO algorithms.

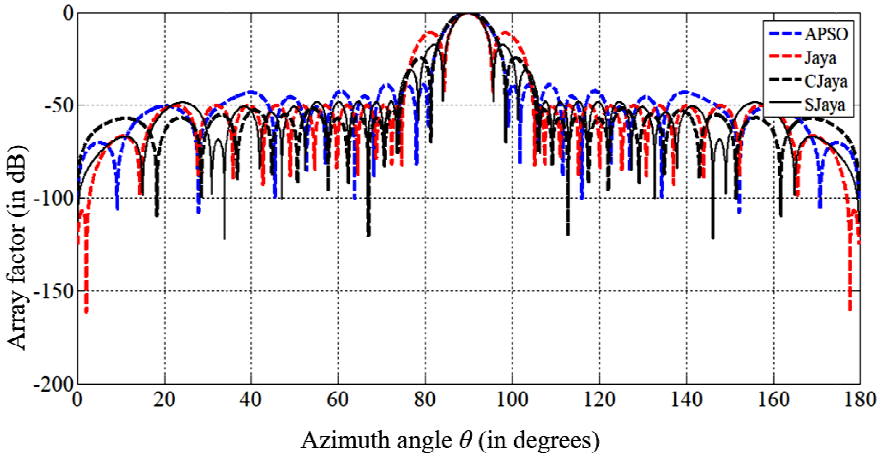

Fig. 10.

The time taken by Jaya, CJaya and SJaya algorithms is 56.309571s, 47.023890s and 49.33760s respectively. It may be observed that CJaya algorithm has taken comparatively smaller time in all the five case studies considered.

4. Conclusions

The synthesis aspects of linear antenna array and dipole antenna array are presented in this paper. Linear array antennas with 28-elements and 37-elements have been synthesized with varying and constant FNBW. The dipole antenna arrays with 12-elements and 20-elements have also been synthesized with varying FNBW. Three different optimization case studies have been formulated for linear antenna arrays and two different case studies are formulated for dipole antenna arrays and the same are optimized using Jaya, SJaya and CJaya algorithms. The results of Jaya, SJaya and CJaya algorithms are compared with those of other algorithms such as PSO, APSO, CSO, CMCSO, HSDEA, DDE, IGA and MGA. It is observed that the Jaya, SJaya and CJaya algorithms have achieved comparatively better results with a high rate of convergence. The CJaya and SJaya algorithms showed comparatively better performance than Jaya algorithm.

In the first case study, synthesis of 28-element linear antenna array is considered with varying FNBW. It is observed that the array pattern suggested by CJaya algorithm achieves a greater SLL suppression as compared to the conventional array pattern, PSO optimized array pattern, CSO optimized array pattern, Jaya optimized array pattern and SJaya optimized array pattern.The CJaya algorithm achieves comparatively a lowest value of FNBW. In the second case study, synthesis of 28-element linear antenna array is considered with constant FNBW. It is observed that the array pattern suggested by SJaya algorithm achieves a greater SLL suppression as compared to the conventional array pattern, PSO optimized array pattern, CSO optimized array pattern and Jaya optimized array pattern. CJaya's performance is next to SJaya algorithm in this case. In the third case study, synthesis of 37-element linear antenna array is considered with varying FNBW. It is observed that the array pattern suggested by CJaya algorithm achieves a greater SLL suppression as compared to the array patterns given by CMCSO, HSDEA, DDE, IGA, MGA, Jaya and SJaya algorithms.

In the fourth case study, synthesis of 12 element dipole antenna is considered with varying FNBW. The CJaya algorithm obtained a better SLL, a precise DRR value and a better fitness value as compared to the APSO and Jaya algorithms, without exceeding the allowable number of generations. The SJaya algorithm has obtained better SLL, FNBW and fitness compared to the CJaya algorithm in this case study. In the fifth case study, synthesis of 20 element dipole antenna is considered with varying FNBW. The CJaya algorithm obtained a lower SLL as compared to the APSO , Jaya and SJaya algorithms. The CJaya algorithm obtained a better fitness function value compared to APSO, Jaya and SJaya algorithms which shows that the CJaya algorithm obtained a better trade-off between SLL and DRR compared to the other algorithms. Overall, out of the 5 case studies considered, CJaya algorithm has provided better results for 3 case studies and SJaya algorithm for 2 case studies.

The results show that the Jaya, SJaya and CJaya algorithms can effectively synthesize a linear array antenna and dipole antenna array with varying topologies, while maintaining the antenna parameters within acceptable limits. The Jaya, SJaya and CJaya algorithms are relatively simple in implementation and do not have any algorithm-specific parameters. The use of chaotic random numbers as scaling coefficients in the CJaya algorithm has successfully enhanced the performance of the algorithm compared to the Jaya algorithm. The CJaya algorithm has good local and global search abilities with high convergence behavior and it may be extended to solve

the complex optimization problems in the electromagnetic engineering domain. However, to improve the algorithm performance further, a better response surface search such as involving transfer of data between random best and worst solutions will be attempted in the near future.

Acknowledgement

The authors acknowledge the editor and the anonymous reviewers for making constructive suggestions to improve the quality of the work. The support extended by Prof. R. V. Rao and Mr. Dhiraj Rai of S.V.National Institute of Technology, Surat, india is also gratefully acknlowledged.

References

[1] Zhang Z, Li T, Yuan F, Yin L. Synthesis of linear antenna array using genetic algorithm to control side lobe level. Comput. Eng. Network. 2014; 277:39–46.

[2] Cen L, Yu ZL, Ser W. Linear aperiodic array synthesis using an improved genetic algorithm. IEEE Trans. Antennas Propag. 2012; 60:895–902.

[3] Chen KS, He ZS, Han CL. A modified real GA for the sparse linear array synthesis with multiple constraints. IEEE Trans. Antennas Propag. 2006; 54:2169–73.

[4] Lin C, Qing AY, Feng QY. Synthesis of unequally spaced antenna arrays by using differential evolution. IEEE Trans. Antennas Propag. 2010; 58:2553–61.

[5] Zhang F, Jia W, Yao M. Linear aperiodic array synthesis using differential evolution algorithm. IEEE Antennas Wireless Propag. Lett. 2013; 12:797–800.

[6] Khodier M, Al-Aqeel M. Linear and circular array optimization: a study using particle swarm intelligence. Prog. Electromagn. Res. 2009; 15:347–73.

[7] Bhattacharya R, Bhattacharyya TK, Garg R. Position mutated hierarchical particle swarm optimization and its application in synthesis of unequally spaced antenna arrays. IEEE Trans. Antennas Propag. 2012; 60:3174–81.

[8] Rajo LE, Quevedo TO. Linear array synthesis using an ant colony,optimization based algorithm. IEEE Trans. Antennas Propag. 2007; 49:70–79.

[9] Karimkashi S, Kishk AA. Invasive weed optimization and its features in electro-magnetics. IEEE Trans. Antennas Propag. 2010; 58:1269–78.

[10] Guney K, Basbug S. Interference suppression of linear antenna arrays by amplitude-only control using a bacterial foraging algorithm. Prog. Electromagn. Res. 2008;79:475–97.

[11] Zaman MA, Matin MA. Non-uniformly spaced linear antenna array design using firefly algorithm. Int. J. Microwave Sci. Tech. 2012, http://dx.doi.org/ 10.1155/2012/256759.

[12] Singh U, Kumar H, Kamal TS. Linear array synthesis using biogeography based optimization. Prog. Electromagn. Res. 2010; 11:25–36.

[13] Saxena P, Kothari A. Ant Lion Optimization algorithm to control side lobe level and null depths in linear antenna arrays. AEÜ Int. J. Electron. Commun. 2016; 70:1339-49.

[14] Pappula L, Ghosh D. Linear antenna array synthesis using cat swarm optimization. AEÜ Int. J. Electron. Commun. 2014; 68:540-49.

[15] Pappula L, Ghosh D. Synthesis of linear aperiodic array using Cauchy mutated cat swarm optimization. AEÜ Int. J. Electron. Commun. 2017; 72:52-64.

[16] Rao RV. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. 2016; 7:19-34.

[17] Wang L, Zhang Z, Huang C, Tsui KL. A GPU accelerated parallel Jaya algorithm for efficiently estimating Li-ion battery model parameters. Appl. Soft Comput. 2018; 65:12-20.

[18] Warid W, Hizam H, Mariun N, Wahab NIA. Optimal power flow using the Jaya algorithm. Energies 2016; 9:678-688.

[19] Singh SP, Prakash T, Singh VP, Babu MG. Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm. Eng. Appl. Artif. Intell. 2017; 60:35–44.

[20] Huang C, Wang L, Yeung RS, Zhang Z, Chung HSH, Bensoussan A. A prediction model guided Jaya algorithm for the PV system maximum power point tracking. IEEE Trans. Sustainable Energy 2018; 9(1):45-55.

[21] Wang L, Huang C. A novel elite opposition-based Jaya algorithm for parametric estimation of photovoltaic cell models. Optik-Int. J. Light Electron Optics 2018; 155:351-356.

[22] Rao RV, More KC. Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm. Energy Convers. Manage. 2017; 140:24-35.

[23] Rao RV, More KC. Optimal design and analysis of mechanical draft cooling tower using an improved Jaya algorithm. Int. J. Refri. 2017; 82:312-324.

[24] Rao RV, Saroj A. A self-adaptive multi-population Jaya algorithm for engineering optimization. Swarm Evol. Comput. 2017; 37:1-26.

[25] Warid W, Hizam H, Mariun N, Wahab NIA. A novel quasi-oppositional modified Jaya algorithm for multi-objective optimal power flow solution. Appl. Soft Comput. 65:360-373.

[26] He D, He C, Jiang L-G, Zhu H-W, Hu G-R. Chaotic characteristics of a one-dimensional iterative map with infinite collapses. IEEE Trans. Circuits Syst. I: Fund. Theory Appl. 2001; 48(7):900-906.

[27]. Saremi S, Mirjalili S, Lewis A. Biogeography-based optimisation with chaos. Neural Comput. Appl. 2014; 25(5):1077-1097.

[28] Florence PV, Raju GSN. Optimization of linear dipole antenna array for sidelobe reduction and improved directivity using APSO algorithm. IOSR J. Electron. Commun. Eng. 2014; 9(6):17-27.

Declarations of interest: None

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.